フル GPU 処理による非圧縮性流体計算

小川　慧 [1)], 青木　尊之 [2)]

1)東京工業大学大学院, 2)東京工業大学学術国際情報センター

1)ogawa@sim.gsic.titech.ac.jp, 2)taoki@gsic.titech.ac.jp

GPU は SIMD 型の高性能アクセラレーターとして HPC アプリケーションを高速に実行することができる。代表的な CFD アプリケーションである非圧縮性流体の物体周りの流れの計算対し、全ての計算を GPU 上で行うことにより VRAM とホスト上のメモリ間通信を不要とし、GPU 本来の演算性能を引き出すことができる。数値計算手法については、計算精度を落とすこと無しに効率的な手法を選択し、圧力の Poisson 方程式はマルチグリッド SOR 法を用いている。グラフィクス・ボード nVIDIA 8800GTS(G92)を用いて CUDA でプログラミングし、1024×512 格子で計算結果をリアルタイム表示できるほど高速に CFD 計算を行うことができた。

1　はじめに

最近、GPU の持つ高速浮動小数点演算機能と高いメモリバンド幅が注目され、GPU が Graphics 処理以外に使われ出している。nVIDIA 社の CUDA[1]がリリースされて以来、さらに SIMD アクセラレーターとして使用することが容易になり、HPC 分野では天体物理における N 体計算[2]、分子動力学[3][4]での計算例がある。流体計算についての報告[5][6]もなされているが、広く使われている格子法の非圧縮性流体に対して高速計算を達成した報告はない。CFD 計算はメモリ参照がスパースで、メモリの読み書きに対して演算量が少ないため、他の SIMD 型アクセラレーター(ClearSpeed, GRAPE)では十分な性能が得られない。

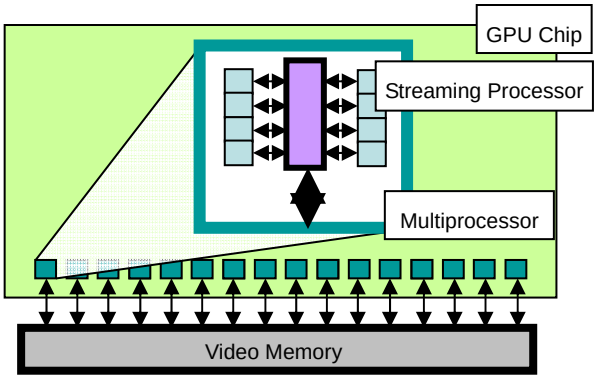

Fig.1 Architecture of CUDA-ready GPU

本計算では前述の CUDA を使用する GPU のアーキテクチャを Fig.1 に示す。Streaming Processor (以下、SP)と呼ばれる最小の計算ユニットがあり、SP が 8 つで一つの Multiprocessor(MP)となり、16Kbyte の高速な Shared メモリを共有する。Global メモリと呼ばれるものが通常の VRAM であり、全ての thread から共有できる。100 個以上の SP が数千～数万 thread を効率的に並列実行する。

GPU を使って HPC アプリケーションを実行する場合、GPU で計算の一部のホットスポットのみを GPU で計算させる方法と、全てを GPU で計算させる方法がある。前者はホスト側のメモリと GPU ボード上のメモリ間の通信がネックになり、アプリケーションの加速率はせいぜい数 10%～数倍に留まる。GPU ボード上で全ての計算を完結させることにより GPU の演算性能を十分に引き出すことができ、CPU での計算に比べて数 10 倍以上高速に計算できる可能性がある。

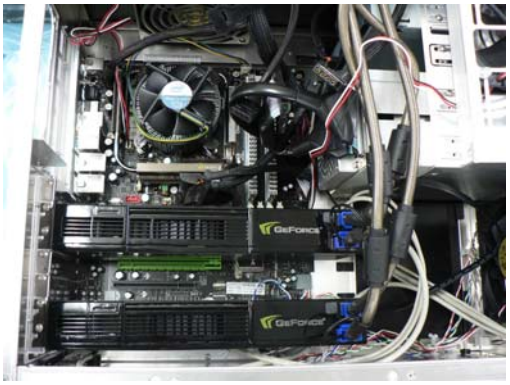

Fig.2 nVIDIA GPU board for HPC Applications.

本研究は CUDA により GPU 上にだけメモリを確保し、完全に GPU だけで実行する CFD アプリケーションを開発し、物体周りの 2 次元非圧縮性流体計算をリアルタイムな表示させながら実行する。GPU としては、ELSA GRADIAC988 GTS 512MB)、ホスト側には Xeon E5420 (2.5GHz) ×2, メモリ 4GByte, CentOS5.1, CUDA1.1 を用いた。

2　非圧縮性流体の計算

日常的な流体現象や工業上の設計・製造過程で重要となる流体解析は非圧縮性 Navier-Stoker 方程式に基づく。

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + \nu\Delta\boldsymbol{u} \qquad (1)$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad (2)$$

u は流速、p は圧力、ν は動粘性係数、レイノルズ数は $\mathrm{Re} = LU/\nu$ である。高精度かつ低計算

負荷の数値計算手法が求められるため、SMAC (Simplified Maker-and-Cell)法に基づいたセミ陰解法とし、圧力勾配項以外は Fractional Step 法の時間 1 次精度で解くことにした。ただし移流項については時間空間 3 次精度の Cubic セミラグランジュ法を Directional Splitting 法で解く。Fractional Step 法に基づいて各パートを以下に述べる。

2.1 移流項計算

Cubic セミラグランジュ法により x 方向の移流計算を GPU で効率的に行うためには、Global メモリへのアクセスを可能な限り少なくする必要がある。CUDA の block を(nx, 1, 1)にとると、各 thread がロードする n ステップの速度を GPU の shared メモリ上に置き cache 的に用いることにより、隣接格子点の参照に対して Global メモリへのアクセスを回避できる。これにより x 方向の移流計算の速度は 56GFLOPS に達する。CPU で同じ計算をした場合、Xeon 2.5GHz で 1 GFLOPS 程度の実行速度であるので、GPU を用いることにより約 50 倍以上高速な計算を行うことができる。

y 向の移流計算は block(1,ny,1)ととると、Global メモリへのアクセスが不連続になり、データ転送速度が極端に低下してしまう。そこで、block(16, 16, 1)などと取り、x 方向の 16 格子点を Global メモリに対して連続にアクセスさせる。y 方向にも 16 格子点あるため、shared メモリの効果が十分に現れ、演算速度は 40GFLOPS を超える。

2.2 拡散項計算

2 次精度中心差分法を用いると、計算ステンシルは x 方向と y 方向のそれぞれ方向の隣接点を参照する。block としては(nx,1,1)とするが、shard メモリは nx のサイズの 1 次元配列を 3 個用意する。jx, jy 格子点を計算する thread は、x 方向の隣接点アクセスは移流計算と同じように行うが、y 方向については jy+1 のデータを格納する shard メモリと jy–1 を格納する shard メモリにアクセスすることにより、Global メモリへのアクセスを回避できる。jy 列の計算が終わり、jy+1 列目の計算を行うとき、先ほど使った jy–1 を格納していた shared メモリの内容は不要になるため、新たに jy+2 列目の Global メモリの値を格納するために使うことにより、jy+1 列目の格子を計算することができる。このように shared メモリをリサイクルすることにより、16kB と少ないサイズでも大きなサイズの Global メモリを扱うことができる。

2.3 マルチグリッド法による Poisson 方程式解法

通常の CPU での計算においても低波数誤差を効率的に低下させる方法としてよく知られるマルチグリッド法を GPU のコードとして実装した。本研究は単相の流体計算であるため、定常反復法として SOR 法を用いた。ただし、thread 間は完全な並列計算であるため、Red & Black 法を用いている。最終段の粗い格子に近づくにつれて格子点数が少なくなるために並列度が低下し、GPU の演算性能としては 20 数 GFLOPS～10 数 GFLOPS となってしまう。Poisson 方程式の計算についても通常の CPU では 1GFLOPS 以下の計算速度しか出ないので、GPU での加速は 10 数倍～20 数倍に達している。GPU は単精度計算であるために残差を十分に評価できないが、マルチグリッド法の 2 段目以下の修正量に対する計算は単精度で十分であり、収束性には影響がない。今回の流体計算で必要とされる精度に対しては、全て単精度で計算しても問題はない。

3 物体周りの流れの計算

CFD の典型的な適用例である 2 次元円柱周りの流れ計算に適用した結果（計算領域の一部）を Fig.3 に示す。格子点数は 1024×512 であり、レイノルズ数を 3000 としている。実際には 3 次元性が出てくるので 2 次元計算は意味が無いが、境界層の剥離および円柱後方のカルマン渦が高精度に計算されていることが分かる。通常はポスト処理により可視化を行うが、GPU で計算することにより、まるで動画を見るようにリアルタイムに計算結果を確認することができる。

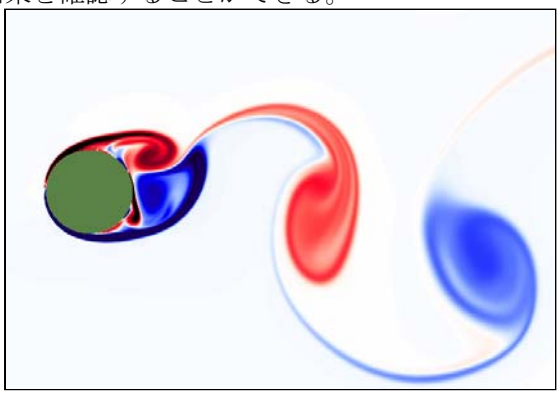

Fig. 3 Incompressible flow around a circle in association with Karman Vortex(Re=3000)

謝辞：本研究の一部は科学技術振興機構 CREST「ULP-HPC:次世代テクノロジーのモデル化・最適化による低消費電力ハイパフォーマンスコンピューティング」及び日本学術振興会 Global COE プログラム「計算世界観の深化と発展」から支援を受けている。

参考文献

[1] nVIDIA,CUDA, http://www.nvidia.com/cuda/
[2] Tsuyoshi Hamada et al., the international conference for high performance computing, networking, storage and analysis, SC08, 2008 (submitted)
[3] 坂牧隆司 他, 第 21 回分子シミュレーション討論会, pp.84-85、2007
[4] John E. Stone et al., Journal of Comp. Chem., vol.28, 16、pp.2618-2640, 2007
[5] Jos Stam, pp.121-128, SIGGRAPH, 1999
[6] John Michalakes et al., 2008 Workshop on Large-Scale Parallel Processing (LSPP), 2008